

# The Study of Quasi Monte Carlo in the Parallel Computation of Invariant Measures

Zizhong J. Wang  
Department of Math/CS  
Virginia Wesleyan College  
Norfolk, VA, USA

Huiqing H. Yang  
Department of Math/CS  
Virginia State University  
Petersburg, VA, USA

Jiu Ding  
Department of Mathematics  
University of S. Mississippi  
Hattiesburg, MS, USA

*Abstract - For a non-singular multi-dimensional mapping  $S: X \rightarrow X$ , the corresponding Frobenius-Perron operator is  $P$ . In the paper, the schemes for generating the quasi-random numbers are studied for the parallel computation for the fixed density of  $P$ . The numerical results for these schemes are presented.*

Keywords: Invariant measures, multi-dimensional dynamic systems, parallel computing, quasi Monte Carlo, quasi-random numbers

## 1. Introduction

Let  $(X, \Sigma, \mu)$  be a  $\sigma$ -finite measure space,  $S: X \rightarrow X$  be a non-singular mapping, and  $A$  be a subset of the space  $X$ . When  $n$  approaches infinity, the following frequency is called the *time mean*:

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \chi_A(S^k(x)) \quad (1)$$

From the Birkhoff ergodic theorem, we know that the *time mean* in (1) is equal to the *space mean*  $\mu(A)$  if  $\mu$  is an ergodic invariant probability measure under  $S$ . That is:

$$\mu(A) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=0}^{n-1} \chi_A(S^k(x)) \quad (2)$$

The invariance of  $\mu$  with respect to  $S$  is that  $\mu(S^{-1}(B)) = \mu(B)$  for all measurable subset  $B$  and the ergodicity means that  $S^{-1}(B) = B$  implies that  $\mu(B) = 0$  or 1.

From (2), we conclude that the time mean in (1) is independent of the initial choice of  $x$  and equal to a constant  $\mu(A)$ , the probability measure of  $A$ .

We introduce the concept of Frobenius-Perron Operator. Let  $P: L(X) \rightarrow L(X)$  be the Frobenius-Perron operator associated with  $S$ , that is defined by:

$$\int_A Pf d\mu = \int_{S^{-1}(A)} f d\mu \quad \forall A \in \Sigma \quad (3)$$

Here the nonnegative function  $f$  is a density. It is not difficult to prove that a fixed density  $f^*$  of  $P$  ( $Pf^* = f^*$ ) presents an absolutely continuous  $S$ -invariant probability measure:

$$\mu_{f^*}(A) = \int_{S^{-1}(A)} f^* d\mu \quad \forall A \in \Sigma \quad (4)$$

For the one-dimensional case,  $S: [0, 1] \rightarrow [0, 1]$ , Ulam proposed a piecewise constant approximation method to calculate the fixed density  $f^*$  of  $P$  [1]. He conjectured that the piecewise constant density will converge to  $f^*$ , that was partially proved by Li [2].

In the next section, we briefly introduce the Ulam's method and the quasi Monte Carlo implementation. The different schemes to construct and generate the random numbers for the parallel computation for the fixed density of  $P$  are studied in the last section. The numerical results are also presented.

## 2. Ulam's Method and Quasi Monte Carlo

Let's consider the multi-dimensional case. Divide  $X$  into  $n$  regular partitions  $I_1, I_2, \dots, I_n$  and denote  $f$  to be a piecewise constant density such that the probability of  $I_i$  is  $a_i$ . Let  $Pf$  be approximated by a piecewise constant density with the probability of  $I_i$  to be  $b_i$ . With the definition of the Frobenius-Perron Operator (3), we obtain:

$$\begin{aligned}
b_i &= \int_{I_i} P f dx = \int_{S^{-1}(I_i)} f dx \\
&= \int_{S^{-1}(I_i) \cap I_1} f dx + \int_{S^{-1}(I_i) \cap I_2} f dx + \dots + \int_{S^{-1}(I_i) \cap I_n} f dx \\
&= \int_{S^{-1}(I_i) \cap I_1} a_1 dx + \int_{S^{-1}(I_i) \cap I_2} a_2 dx + \dots + \int_{S^{-1}(I_i) \cap I_n} a_n dx \\
&= \frac{m(S^{-1}(I_i) \cap I_1)}{m(I_1)} a_1 + \frac{m(S^{-1}(I_i) \cap I_2)}{m(I_2)} a_2 \\
&\quad + \dots + \frac{m(S^{-1}(I_i) \cap I_n)}{m(I_n)} a_n \quad (5)
\end{aligned}$$

Here  $m$  is the Lebesgue measure on  $X$ . Let  $a = [a_1, a_2, \dots, a_n]$ ,  $b = [b_1, b_2, \dots, b_n]$  and the  $n \times n$  matrix  $P_n = [p_{ij}]$  with each entry defined by

$$p_{ij} = \frac{m(S^{-1}(I_i) \cap I_j)}{m(I_j)} \quad (6)$$

Then we have  $b = aP_n$ . Since  $\sum_{j=1}^n p_{ij} = 1$ ,  $P_n$  is a nonnegative and stochastic matrix. According to Frobenius-Perron theorem for nonnegative matrices, there should be a nonnegative vector  $c^T = (c_1, c_2, \dots, c_n)$  with  $\sum_{j=1}^n c_j = 1$ , such that

$$(c_1, c_2, \dots, c_n) P_n = (c_1, c_2, \dots, c_n) \quad (7)$$

Then the piecewise constant density is

$$f_n(x) = \sum_{i=1}^n \frac{c_i}{m(I_i)} \chi_{I_i}(x) \quad (8)$$

that will be the approximation of the fixed density  $f^*$  in (6) and  $P_n$  is called the Ulam's matrix. The above scheme is called the Ulam's method.

To implement the Ulam's method by evaluating the entry of the matrix  $P_n$ , the inverse mapping  $S^{-1}$  in (4) has to be calculated, that may not be feasibly realized for general cases. A quasi Monte Carlo scheme is employed to overcome the difficulty [3]. The basic idea is: generate  $K$  quasi-random numbers  $\{z_{i,k} |_{k=1}^K\}$  in  $I_i$  of  $X$ , and assume  $q_{ij}$  is

the number of the points  $\{S(z_{i,k}) |_{k=1}^K\}$  fallen in  $I_j$ . Then  $p_{ij}$  in (6) can be approximated by

$$p_{ij} \approx \frac{q_{ij}}{K} \quad (9)$$

The implementation procedure for the above Ulam's method and quasi Monte Carlo can be concluded as follows:

- Set the starting point  $c^T = (1, 1, \dots)$ .
- Evaluate  $d^T = c^T P_n$  to get  $d^T = (d_1, d_2, \dots, d_n)$ .
- Compute the error  $E = \|d - c\|_1$  using 1-norm.
- Move  $d$  to  $c$  and repeat (b) and (c) until  $E < e$ , where  $e$  is a pre-defined constant, such as 0.1 or 0.01.
- The vector  $c$  will represent the coefficients of the fixed density approximation in (8).

### 3. Quasi-random Numbers and Parallel Computing

For the one-dimensional case, the distribution of the quasi-random numbers is unique, that is, the uniformly partition for the interval  $I_i$ . But there are two schemes to partition the interval for generating the random numbers:

- Left-to-right ( $\rightarrow$ ) partition. Assume the interval is  $[i\_left, i\_right]$  and there are  $M$  quasi-random numbers are generated. The  $k$ th quasi-random number in C code is:

$$R[k] = i\_left + k * h / M;$$

Here,  $h = i\_right - i\_left$ , that is the width of the interval.

- Right-to-left ( $\leftarrow$ ) partition. The corresponding  $k$ th quasi-random number in C code is:

$$R[k] = i\_right - k * h / M;$$

For the dynamic system:  $S(x) = 4x(1-x)$ , that is the so-called "logistical model", the fixed density is:

$$f^*(x) = \frac{1}{\pi \sqrt{x(1-x)}} \quad (10)$$

The numerical results are shown in Table 1. Here, the  $L^1$ -error is defined as the  $L^1$ -norm for the difference between the exact density ( $f^*$ ) and its approximation ( $f_n$ ):

$$L^1\text{-error} = \|f^* - f_n\|_1$$

The total number of partitions in  $[0, 1]$  is denoted as  $n$ . The number of the quasi-random numbers is  $M = 2000$ .

From Table 1, we can see that the two schemes do not have significant differences.

Table 1  $L^1$ -error Comparisons

$n$	(a)	(b)
16	0.359513	0.359355
32	0.293488	0.293380
64	0.249365	0.249274
128	0.214782	0.214704
256	0.179664	0.179605

The above experiment is performed on the Linux server at Virginia Wesleyan College.

For the multi-dimensional case, similar to the 1-D case, the distribution of the quasi-random numbers can also be the uniformly portioning for the regular shape  $I_i$ . But there are many schemes to partition the regular shape.

Let's consider the 2-D case, the regular shape is a square. Assume we will produce 9 quasi-random numbers (sub-squares) as in Figure 1.

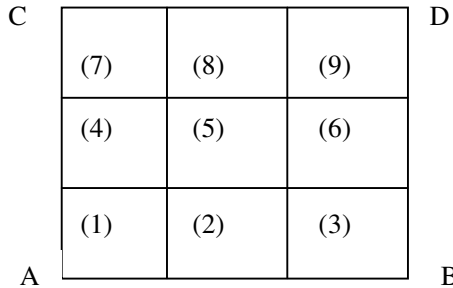


Figure 1

Here, the coordinates of the points are: A ( $i_{x\_left}, i_{y\_left}$ ), B ( $i_{x\_right}, i_{y\_left}$ ), C ( $i_{x\_left}, i_{y\_right}$ ), D ( $i_{x\_right}, i_{y\_right}$ ).

For simple reason, in this paper we consider the following two schemes to partition the square for generating the quasi-random numbers (sub-squares).

(i) Row-by-row partition (Figure 1). The corresponding C code implementation is:

```
for(k=0; k<M; k++)
{
    row(k)=ix_left+k*h/M;
    for(l=0; l<M; l++)
    {
        col(l)=iy_left+l*h/M;
        ...
    }
}
```

(ii) Column-by-column partition (Figure 2).

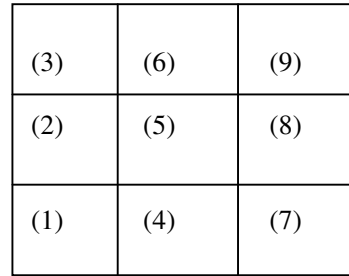


Figure 2

The corresponding C code implementation is:

```
for(l=0; l<M; l++)
{
    col(l)=iy_left+l*h/M;
    for(k=0; k<M; k++)
    {
        row(k)=ix_left+k*h/M;
        ...
    }
}
```

As described in [3], the quasi Monte Carlo is a time-consuming process and the parallel computing is employed to improve the process, especially for the multi-dimensional systems.

Let's consider the following 2-D dynamic system [4]:

$$S(x, y) = (T, R). \quad (11a)$$

Where  $T$  and  $R$  are defined as follows.

$$T(x) = \begin{cases} 2x & 0 \leq x < \frac{1}{2} \\ 2(1-x) & \frac{1}{2} \leq x \leq 1 \end{cases} \quad (11b)$$

$$R(y) = \begin{cases} \frac{2y}{1-y} & 0 \leq x < \frac{1}{3} \\ \frac{1-y}{2y} & \frac{1}{3} \leq x \leq 1 \end{cases} \quad (11c)$$

The corresponding fixed density is given by:

$$f^*(x, y) = \frac{2}{(1+y)^2} \quad (12)$$

The numerical results comparing the above two partition schemes are included in Table 2.

The experiments were performed on the Origin 2800 at the Mississippi Center of Supercomputing Research. The supercomputer is now equipped with 128 processing units (CPUs), 64 gigabytes of memory, and 1.6 Terabytes of disk.

Table 2  $L^1$ -error Comparisons

$n \times n$	(i)	(ii)
4x4	0.110288	0.165888
8x8	0.052194	0.155943
16x16	0.031428	0.152672
32x32	0.015245	0.150425

From the above results, we see that the two schemes have significant differences. Actually, the row-by-row partition scheme works perfectly while the column-by-column partition scheme fails to decrease the error significantly with the increment of the partitions. Further experiments indicate that the errors for the column-by-column scheme will increase with the increment of the processors. A possible reason for this phenomenon is because we employ the row partition for the parallel computation of the Ulam's matrix  $P_n$ .

In conclusion, for the multidimensional dynamic systems, the quasi-random numbers are the regular sub-shapes for the evaluation of the Ulam's matrix  $P_n$ . The numerical results show that the row-by-row partition scheme fully outperforms the column-by-column scheme. A further study is needed to conduct the related efficiency analysis.

*Acknowledgement* Computer time grant from the Mississippi Center of Supercomputing Research is kindly acknowledged.

#### 4. References

- [1] S. Ulam. A Collection of Mathematical Problems, Interscience Tracts in Pure and Applied Math., 8, Interscience, New York, 1960.
- [2] T. Y. Li. Finite approximation of Frobenius-Perron operators. A solution to Ulam's conjecture. *J. Approximation Theory*, 17:177-186, 1976.
- [3] J. Ding and Z. Wang. Parallel computation of invariant measures. *Annals of Operations Research*, 103, 283-290, 2001.
- [4] J. Ding and A. Zhou. Finite approximation of Frobenius-Perron operators. A solution of Ulam's conjecture to multi-dimensional transformations. *Physica*, D 92, 61-68, 1996